Linear Regression

Nikhil Jangamreddy (2018csm1011)

February 4, 2019

Abstract

In this experiment we studied Linear Regression, which is linear model for modelling continuous scalar output. Linear regression can be solved using two approaches namely Gradient descent and Closed form solution. We used Closed form solution to predict weights based on Training data and analyse its performance on Testing data. Apart from these various experiments based on Average Testing MSE, λ, Fraction values are done to understand impact of these parameters on getting a better fitted model.

1 Linear Ridge Regression :

In this experiment, we implement linear regression in order predict a continuous value (age of Abalone). The dataset is collected from UCI repository [1]. Then we perform series of experiments to understand and analyse the Linear Ridge Regression model.

1.1 Encoding the first column into three columns :

Given data set contains 4711 $\times$ 9 columns, out of which there is first column to identify Male, Female and Infant. We try to encode first column as [0,0,1], [1,0,0] and [0,1,0] respectively. Now the dataset will have 4711 $\times$ 11 columns in the data after encoding.

1.2 Standardizing the data :

Before splitting the data into train and test data set we standardise the data . In standardisation the values will be transformed such that mean and variance of each column will be 0 and 1 respectively. During standardisation first three columns as well as last column are not standardised because first three columns are already 0,1s and last column represents Output.

1.3 Implementation of Linear Regression Function :

Linear Ridge Regression can be solved by Gradient descent and Closed form methods. Here, Implementation of Linear Regression Function is done by using closed form solution as described below :

$$W = (X^t X + \lambda I_p)^{-1} X^t Y$$

Where X is Input Matrix, which corresponds to 4711 $\times$ 10 matrix data.
Y is Output Matrix , which corresponds to 4711 $\times$ 1 matrix data.
λ is penalty factor used in case of regularisation.
I_p is Identity matrix , which corresponds to 10 $\times$ 10 matrix data.
We implement mylinearregval function and compute $\mathbf{X} \times \mathbf{W}$ and returns the column matrix which contains the Prediction values.

1.4 Partition into test and train data :

After pre-processing the data, we split the data into train, test and validation data.As per conditions specified, divide 20% of data into test and remaining 80% of data into train and validation data based on train fraction value chosen. We build Linear Regression model based on the Training data and check performance on the test data. Validation data is not used as part of experiments.

Note : Test data input is Standardised using the mean and standard deviation from the Training data.

1.5 Computing mean square error :

After finding regression weights, we try to find the predicted values by using mylinearregval function and find mean square error by using below formula :

$$\text{mean square error} = (\sum (predictedvalue - actualvalue)^2)/n$$

Mean square error is used as standard for checking performance of Linear Regression weights calculated using Training data.

1.6 Effect of λ on Error change, Average mean square error :

Randomly choose data points into train and test data. After standardisation divide the data based on data size fraction .Now for 100 random iterations for each of 10 values of lambda and 10 values of data split fraction calculate mean square error for train and test dataset(not validation data set). Average mean square error for train as well as test is printed after execution of part 1.6.

Chosen Lambda values, fraction values are as follows :

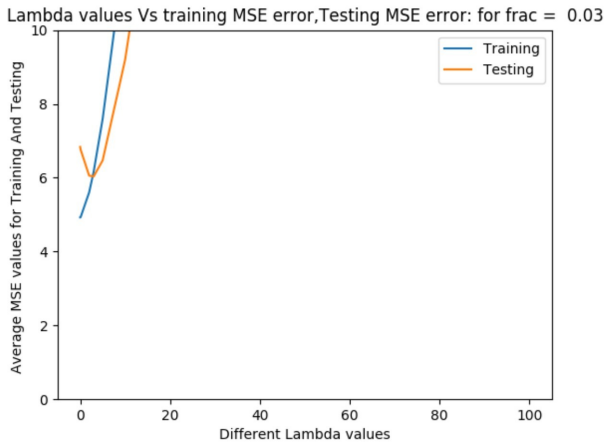
Lambda values = [0.0001,0.01,0.1,2,3,5,10,25,50,100]
Fraction values = [0.1,0.2,0.3,0.4,0.45,0.55,0.6,0.7,0.9,1.0]

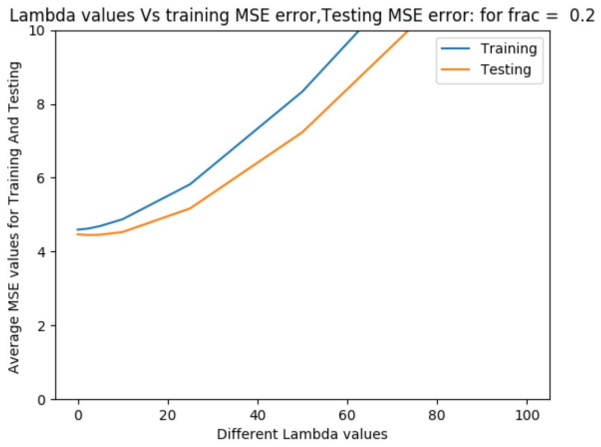
Note : For each value corresponding to Lambda and Fraction value, the train data is randomly chosen, run for 100 iterations and average of 100 iterations is referred as Average training MSE error. Similarly average Testing MSE error is also calculated.

Average Training MSE over 100 iterations for 100 combination of Lambda values and fraction values is **5.59154854**.
Average Testing MSE over 100 iterations is for 100 combination of Lambda values and fraction values is **5.81011069**.

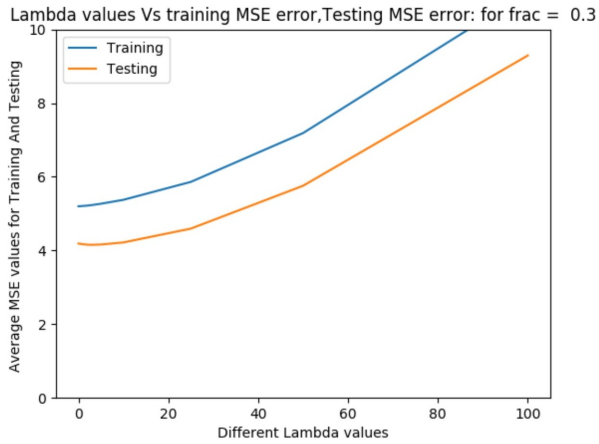
1.7 Effect of Mean Square Error, λ with respect to Fraction value:

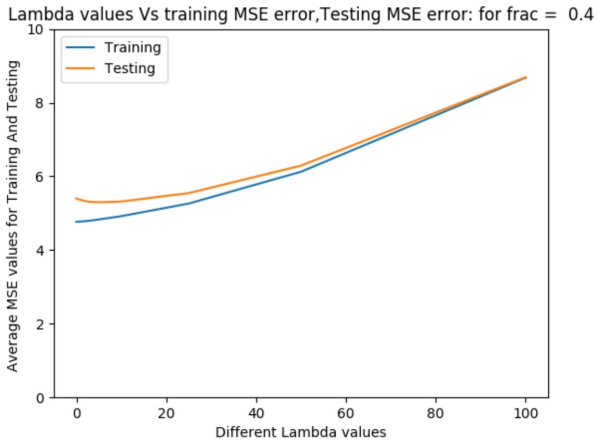
For a particular fraction value, plotting lambda values vs average training MSE error as well as average testing MSE error.



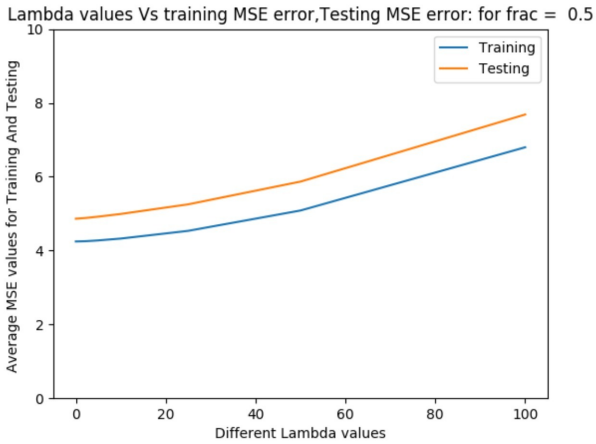


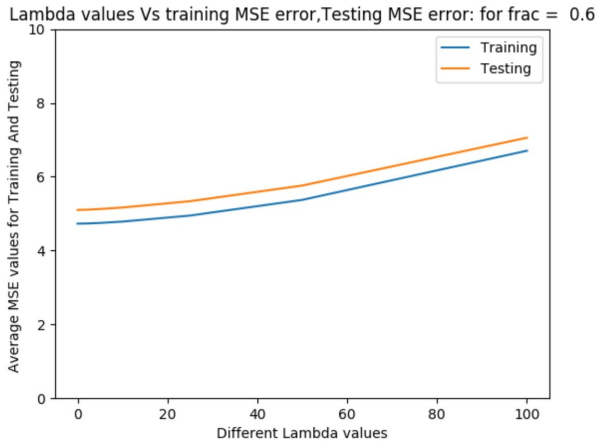
Fraction values = [0.03,0.2]



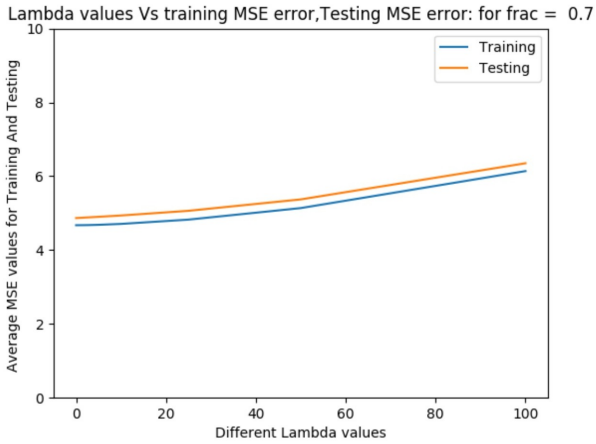


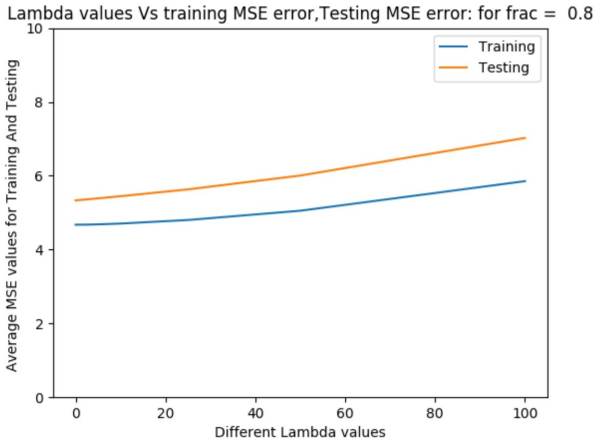
Fraction values = [0.3,0.4]



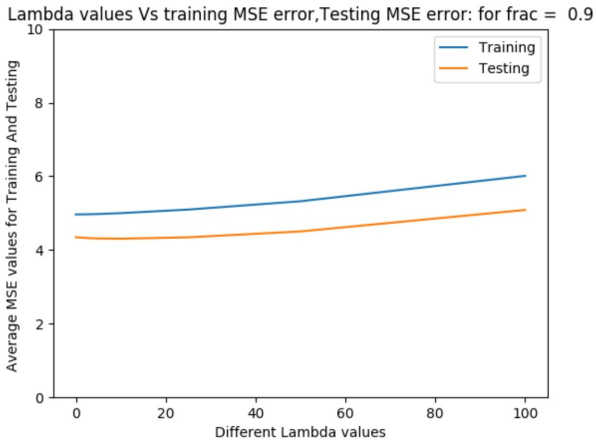


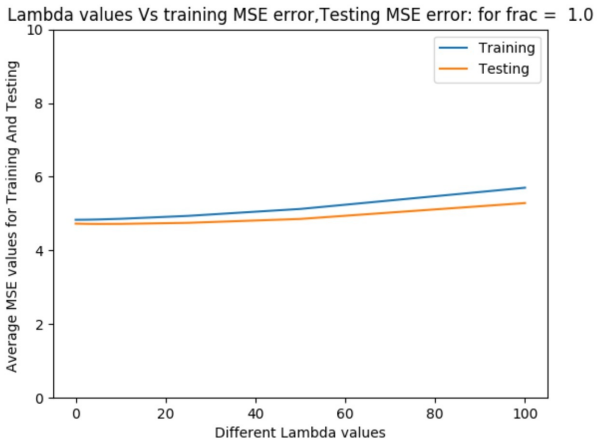
Fraction values = [0.5,0.6]





Fraction values = [0.7,0.8]





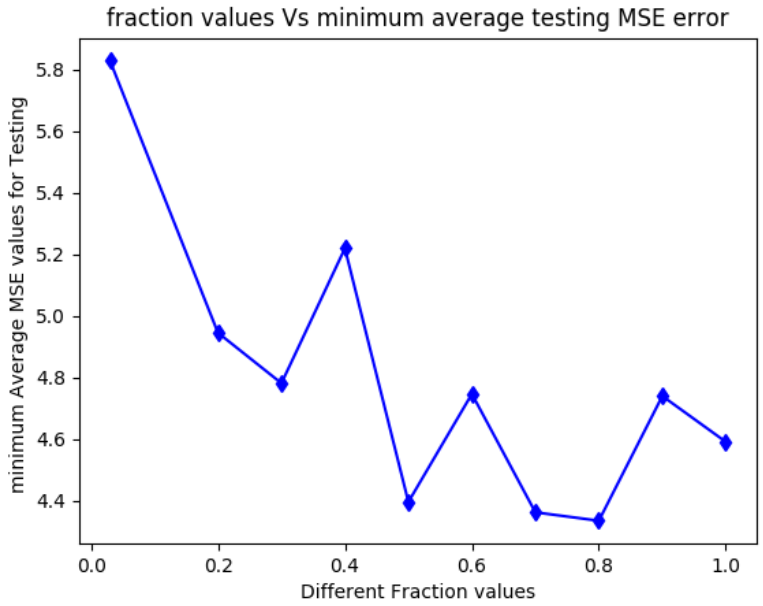
Fraction values = [0.9,1]

1.7.1 Observations :

- For a particular fraction value, As the value of λ value increases. the average training MSE, average Testing MSE is increasing.
- As the fraction values are small, the increase in average mean square training error is rapid which suggests error is increasing at a fast rate, which means model is very unstable. But for the average mean square testing error the graphs still remain to increase rapid for large values of fraction compared to training error values.
- When fraction value increases, average Testing MSE decreases because the data used for training increases and better fitted weights for model will be learnt.
- As training fraction value increases, average training MSE decreases till a particular value then it increases because as training fraction is increasing, number of instances in training data increases which leads to better fit model. After a particular value of fraction, testing error might increase because model might overfit.

1.8 Effect of min Testing MSE, λ with respect to Fraction values:

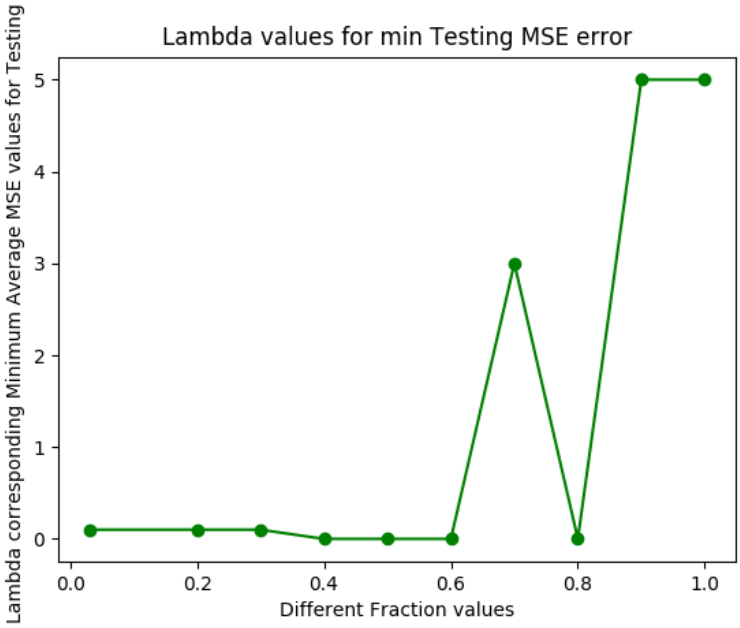
To analyse more, here we plot for different values of Fraction values on X axis, with respect to minimum Average Testing mean square error on Y axis for different λ values.



Fraction values = [0.0001,0.01,0.1,2,3,5,10,25,50,100]

1.8.1 Observations :

- We can observer Minimum testing MSE for different values is corresponding to fraction value = 0.8.
- As training set fraction is lower, minimum testing MSE is higher considering number of data instances being lower leads to poorly fitted weights.
- As training set fraction increases, minimum testing MSE decreases till a particular value suggesting model weights are fitting better as fraction value increases. After a particular fraction value, min Testing MSE starts to increase possibly because of overfitting.



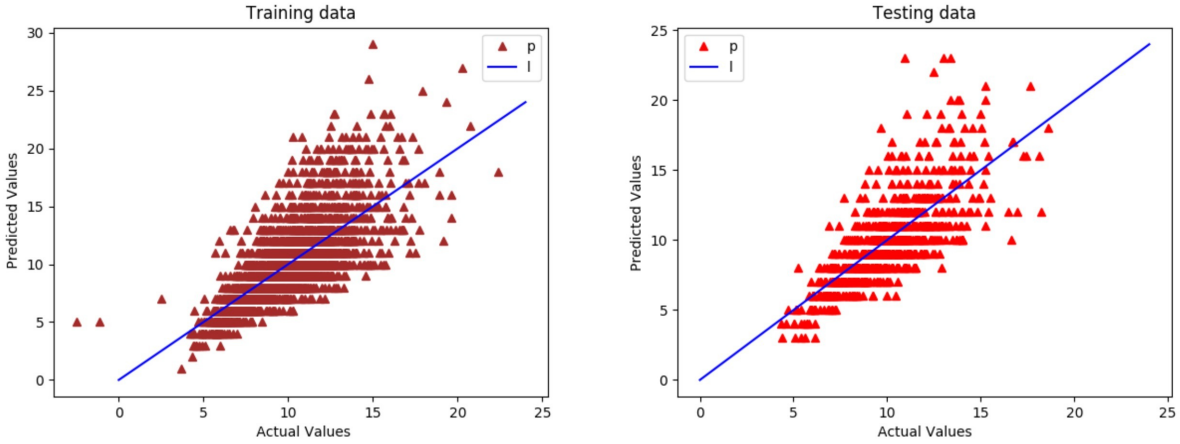
Fraction values = [0.0001,0.01,0.1,2,3,5,10,25,50,100]

1.8.2 Observations :

- We are obtaining minimum testing MSE for maximum value considered which also helps in understanding that we can try even from large values and check testing error for large fraction values as [0.9,1].
- Here for global minimum MSE testing error choose fraction value that gives minimum testing MSE i.e.., **fraction = 0.8** and corresponding λ **= 0.0001.**
- For higher Fraction values λ values are higher, may be model is overfitting and λ values are higher to regularise it.

1.9 Actual Vs Predicted values with respect to y=x line :

To understand the impact of particular values on error, for instance in case of outliers best way is to visualise Actual Vs Predicted value with respect to y=x line. In case the predicted values are close to actual values i.e.., best fit model, values will be clustered closely towards y=x line. Else points will be farther away from the y=x line.



Fraction value = 1, Lambda = 3

1.9.1 Observations :

- Few actual values in test data are very far from y=x line. The reason is due to small set of data i.e.., 4711 rows and output is a continuous value with range. Typically dataset with large number of rows are expected to perform better.
- Majority of the points in Testing data close to the y=x line infers learned weights are performing well which implies model is working well.
- Points in testing data are close to y=x line shows that weight values learnt by training data are not highly overfitted.
- Training data is more approximate to y=x and clustered towards y=x line which is expected result.

References

[1] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.